Objectives:

- Learn how to implement a generic dynamic array in the OOP style.
- Work with predefined tests and header files
- Practice using template concepts and template files.
- Practice using Operator overloading for programmer-defined classes.
- Practice using inheritance to derive new classes.

Project 1: Dynamic Array [10 Points]
Files:

testMain-1.cpp  Download testMain-1.cpp

varList.h  **Download varList.h**

Useful Video (Similar Program to the first project. I would watch the video if I were you)

https://web.microsoftstream.com/video/0521b2fd-5bfe-4133-846c-ab38fa7ed91b (Links to an external site.)

This is a video on how to create template files in Visual Studio:

Watch 'CS 244-002, Data Structures Virtual Class Meeting_TemplateFilesCreation' | Microsoft Stream (Links to an external site.)

Project description:

Write a program that allows the user to add or remove value to a collection of a given data type (primitive or programmer-defined data types such as a class or struct) and then print all of them. A user may enter the value twice in which case, the repeated value should not be entered.

Here is an example program dialog, assuming the list created to hold integer values:

Enter operation [A/R/Q]: A

Enter a value : 10

Current values stored in the list: 10

Enter operation [A/R/Q]: A

Enter a value : 20

Current values stored in the list: 10, 20

Enter operation [A/R/Q]: A

Enter a value : 20

Current values stored in the list: 10, 20

Enter operation [A/R/Q]: R

Enter a value : 10

Current values stored in the list: 20

Enter operation [A/R/Q]: A

Enter a value : 7

Current values stored in the list: 20, 7

Enter operation [A/R/Q]: A

Enter a value : 15

Current values stored in the list: 20, 7, 15

Enter operation [A/R/Q]: R

Enter a value : 7

Current values stored in the list: 20, 15

Enter operation [A/R/Q]: Q

The size of the user input can be arbitrarily large. For this program, to accommodate user input, you need to implement an underlining array whose size varies as necessary. The values in the array should not be sorted.

Note: In this project, you will create two C++ projects, one to implement the missing methods (inside a template file “implementation.template”) and the second to use what has been implemented and tested, in part 1, to write the target program. That is, the same way you did in the battleship game project.

Part 1: Variable Size Array with Classes, Testing.

Create a project titled Testing_VarArray_Components. Implement the dynamically expanding and contracting generic array described above in an OOP style. You should use the header file posted on Canvas (code is given below for your convenience). The class attributes are a pointer to the dynamically allocated array **dAarray** and the array size **listSize**. This class contains two groups of methods:

- Member functions**printList**(), **check**() **addValue**() and **removeValue**() as described below.
- **copy constructor**, **overloaded assignment,** and **destructor** ensure correct handling of the objects with dynamically allocated members.
- **Note: All the class method implementations must be implemented in the "**template" file that must be included in the header file and protected against multi-inclusion.
- **The template file must be protected against multi-inclusion (#ifndef XYZ, #define XYX ..... #endif**

Your code should work with the test posted on Canvas (and provided below). It is designed to test your implementation of varArray class.

- **printList**()prints the arrays' contents since it is a member of the same class.
- **check**()takes a value (key) and checks if the value is in the array. If the value is in the array, returns the index of the element of the array holding the value. Otherwise, returns -1.
- **addValue**()takes and adds a value (of a proper data type) to the array if the value is not already there. Note that the array cannot be enlarged to accommodate the new value. Instead, this function needs to allocate a new temporary array whose size is one more than the size of the old array, copy the contents of the old array to the new one, including the new value, deallocate the old array and then assign the address of the new array back to the original pointer. Notice that since it is a member of the same class., the call to this function results in effectively "enlarging" the array pointed to by the pointer. (remember the BasicVector example 😊)
- **removeValue**()takes a value (key) and removes the value if it is present in the array. If the value (key) is removed, the array shrinks. Note that the function should not change an array in any way if the value (key) is not present in it. If the value is present, then the function should allocate a new array of smaller size and then copy all the values to it, except the one that needs to be erased. After copying is done, deallocate the old array and update the array pointer to point to the new array. Copying and removing one element may be done in a single for-loop.

Part 2: Variable Size Array with Classes, Implementation.

Create a project titled VarArray_Project. Using the class implemented in the first part of this lab, write a program that asks the user to input values to add to and remove from the array and then print its contents. In this project, reuse the header file given to you and the template file that contains the implementation of all of the class methods.

This is a video on how to create template files in Visual Studio:

Watch 'CS 244-002, Data Structures Virtual Class Meeting_TemplateFilesCreation' | Microsoft Stream (Links to an external site.)

Make sure your programs adhere to the proper programming style. Submit your projects to the designated dropbox on Canvas. Please make sure to reach out to your instructor in a timely manner for effective assistance.

Treat this assignment as a learning project not only as an evaluation component.

Project 2: Operator Overloading [10 Points]

In this question, you are expected to fix a program that is described below (next page) so that it works as expected without changing the main function (test).

The program (shown below) has only one class named **Player** that has multiple member variables and methods. The program generates three instances of the class **Player** that are expected to represent soccer players' information gathered from a soccer video game. The collected information is used to compare the players (objects) to know the player of the game. The player of the game is determined based on the following scheme:

1*: The player of the game is the one who has the highest number of goals and assists in that game (summation).*

2: if the summation of the goals and assists that are conducted by two players is the same, then the system compares between them based on the number of effective touches done by the players throughout the game.

For example, if players P1 and P2 have scored 2 goals in the game and done 7 assists each, the system compares their effective touches. If P1 has 27 effective touches and p2 has 32 touches, then P2 will be ranked before P1.

3: if the players being compared have the same number of (goals + assists) and the same number of effective touches, then running distance is used to compare between them such that the player who runs longer is ranked high than the other one.

Example:

If the user chooses **Cristiano Ronaldo**, **Lionel Messi**, and **Gareth Bale** as candidates for the player of the game and their collected information was as shown in the table below, the system will choose (supposed to) Gareth Bale to be the player of the game since his goals and assists (15) is the largest.

Metrics	***Cristiano Ronaldo***	***Lionel Messi***	***Gareth Bale***
Goals + Assists	***4 + 7 = 11***	***2 + 9 = 11***	***1 + 14=15***
Effective Passes	***32***	***37***	***33***
Running Distance	***17***	***13***	***17***

Note that If the other two players are compared, Messi would be ranked higher than Ronaldo because his effective passes number is higher than Ronaldo, although their sums of goals and assists number are the same (11 each).

So, your job is to fix the program shown below using a technique that you learned in class to make the compiler able to compare between objects of the class Player using the relational operator >. That is, try to modify or add something to the class so that the compiler will not throw any compile-time errors (caused by the comparison operation.)

Hints:

- ***Think about a way that would teach the compiler how to treat the operator > if the operands are objects of the class Player.***
- ***Operator overloading examples studied in class.***

```
#include <iostream>

#include <string>

using namespace std;

class Player {

public:

    Player(string _fullName,int _numberOfGoals,int _numberOfAssists,int
_numberOfEffectiveTouches,int _runningDistanceKM) {

        fullName= _fullName;

        numberOfGoals= _numberOfGoals;

        numberOfAssists= _numberOfAssists;

        numberOfEffectiveTouches= _numberOfEffectiveTouches;

        runningDistanceKM= _runningDistanceKM;

    }

    void printInfo() {

        cout<<endl<< fullName<<endl;

        cout << numberOfGoals << endl;

        cout << numberOfAssists << endl;

        cout << numberOfEffectiveTouches << endl;

        cout << runningDistanceKM << endl;

    }

    string fullName;
```

//You code goes here 😊

```
private:
    int numberOfGoals;
    int numberOfAssists;
    int numberOfEffectiveTouches;
    int runningDistanceKM;
};
int main()
{
    Player player1("Cristiano Ronaldo dos Santos",4,7,32,17);
    Player player2("Lionel Andrés Messi",2,9,37,13);
    Player player3("Gareth Frank Bale", 1, 14, 33, 17);
    //binary 'operator' : 'type' does not define this operator or a conversion to a type acceptable to the predefined operator
    if ( player1> player2 && player1> player3)
        cout << "Player of The game is " << player1.fullName << endl;
    else if (player2> player1 && player2> player3)
        cout << "Player of The game is " << player2.fullName << endl;
    else
        cout << "Player of The game is " << player3.fullName << endl;
```

```
    return 0;
}
```

Project 3: Inheritance and Abstract Classes [10 Points]

1. Design and Implement an abstract class called '**Drone'** that has the following members:
    a. A public String variable '**droneName'** (holds the drone name. The default name is "**Unknown Drone**"
    b. A private Boolean variable '**licensed'** (true for licensed drone and false otherwise).
    c. A private Decimal variable '**payload**' (holds the payload of the drone, but it is 1.5 by default.)
    d. Two getters for the members described in 2 and 3.
    e. An **abstract** method (pure virtual) '**void printPayload()**'. (overridable if needed by children classes)
    f. The class has a construct that is used for initializing the class members.
2. Show (design and implement) how derivate a sub-class '**ScientificDrone'** from the class '**Drone'** described in B with two decimal class members '**sensorsWeight'** and '**batteryWeight'**. Then provide a concrete implementation for the method **printPayload** inherited from the superclass '**Drone'** such that it calculates and outputs the actual payload of the drone, based on the values stored in the class members, **sensorsWeight** and **batteryWeight**.

Please make sure that you implement proper constructors as well (initialize the inherited members from the parent and the child members).